Institute for Formal Models and Verification
Johannes Kepler University Linz

JKU
JOHANNES KEPLER
UNIVERSITY LINZ

VL Logik (LVA-Nr. 342208), Winter Semester 2014/2015

Satisfiabiliy Modulo Theories: The SMTLib2 Input Format

Version 2015.1

Armin Biere (biere@jku.at)
Martina Seidl (martina.seidl@jku.at)

SMT-Lib

SMT-Lib (`www.smtlib.org`) is a community portal for people working on and with SMT Solving including

- ... a standard for describing background theories and logics
  6 background theories, $>$ 20 logics
- ... a standard for input/output of SMT solvers
- ... a collection of 95492 benchmark formulas
  totalling 59.2 GB in 383 families over 22 logics
- ... a collection of tools
- ... the basis of the annual competition

SMT Solvers

- aim of an SMT solver: check satisfiability of formula ϕ
  - not over all (first-order) interpretations
  - but with respect to some background theory

- artifacts of an SMT solving system compliant to SMTLib v2:
  - based on many-sorted first-order logic with equality
  - background theory: taken from catalogue of theories
    - basic theories
    - combined theories
  - interface: command language
  - input formula

The SMT-Lib Command Language

- communication with the SMT solver
  - textual input channel
  - two textual output channels
    - regular output
    - diagnostic output
- primary design goal: interaction between programs
- types of commands
  - defining sorts and functions
  - managing assertions
  - checking satisfiability
  - setting options
  - getting information
  - `exit`
- responses: `unsupported`, `success`, `error` ⟨string⟩

Theories and Logics

A theory

- ... defines a vocabulary for sorts and functions (signature).
- ... associates each sort with literals.
- ... may be infinite.
- ... has often an informal specification (in natural language).

A logic

- ... consists of at least one theory.
- ... restricts the kind of expressions to be used.
- ... has often an informal specification (in natural language).

SMTLib provides various theories and logics.

Some Logics without Quantifiers

Logic	*Description*
QF_UF	formulas over uninterpreted functions
QF_LIA	formulas over linear integer arithmetic
QF_NIA	formulas over integer arithmetic
QF_BV	formulas over fixed-size bitvectors
QF_ABV	formulas over bitvectors and bitvector arrays
QF_AUFBV	formulas over bitvectors and bitvector arrays with unint. func.
QF_AUFLIA	linear formulas over integer arrays with uninterpreted functions

Terms, Functions, and Predicates

- Structure of terms and functions:
  - ⟨*constant*⟩
  - ⟨*identifier*⟩
  - `as` `(` ⟨*identifier*⟩ ⟨ *sort* ⟩ `)`
  - `(` ⟨*identifier*⟩ ⟨*term*⟩*+* `)`
  - `(` `as` `(` ⟨*identifier*⟩ ⟨ *sort* ⟩ `)` ⟨*term*⟩*+* `)`
  - quantifier terms with `forall, exists`
  - attributed terms `!`
  - bound terms with `let`

- example `(or (> p (+ q 2)) (< p (- q 2)))`

- terms are always typed
- no syntactic difference between functions and predicates

Declaring Functions (and Constants)

- `declare-fun` $(\sigma_1 \ldots \sigma_n)$ σ:
  - declaration of new function with n parameters of sorts $\sigma_1 \ldots \sigma_n$
  - return value of sort σ
- constants are 0-ary functions

Example

- `(declare-fun x () Bool)`
- `(declare-fun f (Int Int) Bool)`
- `(declare-fun ff ( (Int Int Bool) ) Int)`

Satisfiability Commands

- `(assert` ⟨*term*⟩`)`
  - `term` is of sort `Bool`
  - solver shall assume that `term` is true
- `(check-sat)`
  - check consistency of conjunction of assertions
  - response: `sat, unsat, unknown`
- get a solution with `(get-model)`

Example

```
(set-option :model true)
(declare-fun x () Int)
(assert (>= (* 3 x) (+ x x)))
(check-sat)
(get-model)
```

Example: Boolean Expressions

- Boolean expressions are defined in the *Core Theory*
- sort: `Bool`
- constants: `true, false` (both of sort `Bool`)
- functions:
  - `not`
  - `or, xor, and, =>`
  - `=, distinct` (equality, inequality)
  - `ite` (if-then-else)

Example

```
(set-logic QF_UF)
(declare-fun x () Bool)
(declare-fun y () Bool)
(assert (and (or x (not y)) (or (not x) y)))
(check-sat)
(exit)
```

Example: Real Expressions

- Real expressions are defined in the *Real Theory*
- sort: `Real`
- constants: numerals, decimals (all of sort `Real`)
- functions with signature:
  - (- (Real) Real) ; negation
  - (- (Real Real) Real) ; subtraction
  - (+ (Real Real) Real)
  - (* (Real Real) Real)
  - (/ (Real Real) Real)
  - (<= (Real Real) Bool)
  - (< (Real Real) Bool)
  - (>= (Real Real) Bool)
  - (> (Real Real) Bool)

Example

```
(set-logic QF_LRA)
(declare-fun x () Real)
(declare-fun y () Real)
(assert (and (>= (* 2 x) (+ y 3.2)) (= x y)))
(check-sat)
```

Example: Array Expressions

The theory of Arrays defines functions to read and write elements of arrays.

- sort: `Array` <sort of index> <sort of elements>
- functions
  - (**select** `(array index)` `value`) where
    - `array` is of sort (`Array` <sort of index> <sort of elements>)
    - `index` is of sort <sort of index>
    - `value` is of sort <sort of elements>
  - (**store** `(array1 index value)` `array2`) where
    - `array1`, `array2` are of sort (`Array` <sort of index> <sort of elements>)
    - `index` is of sort <sort of index>
    - `value` is of sort <sort of elements>

Example

```
(declare-fun a () (Array Int Bool))
(declare-fun b () (Array Int Bool))
(assert (= (select a 1 ) true))
(assert (= (store b 1 false) a))
(check-sat) ; result is unsat
```

Example: Fixed-Sized Bitvectors Expressions

- sort: `(_ BitVec n)` where *n* is the size of the bitvector
- functions:
  - (op1 (_ BitVec m) (_ BitVec m))
    - with op1 $\in$ {bvnot, bvneg}
  - (op2 (_ BitVec m) (_ BitVec m) (_ BitVec m))
    - with op2 $\in$ { bvand, bvor, bvadd, bvmul, bvudiv, bvurem, bvshl, bvlshr }
  - (bvult (_ BitVec m) (_ BitVec m) Bool)
    - binary comparison
  - ((_ extract i j) (_ BitVec m) (_ BitVec n))
    - extract contiguous subvector from index i to index j
  - (concat (_ BitVec i) (_ BitVec j) (_ BitVec m))
    - combines two bitvectors

Outlook

SMTLib2 offers many more language concepts, for example:

- Makros
- User-defined sorts
- Many Options
- Scopes
- ...

More infos:

- `http://smtlib.cs.uiowa.edu/papers/smt-lib-reference-v2.0-r12.09.09.pdf`
- `http://www.grammatech.com/resources/smt/SMTLIBTutorial.pdf`